THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 62, No. 1, January 1983
Printed in U.S.A.

The 3B20D Processor & DMERT Operating System:

3B20D Processor Memory Systems

By I. K. HETHERINGTON and P. KUSULAS

(Manuscript received March 17, 1982)

The memory system supplied with the 3B20D Processor provides a high-reliability, high-performance, main-frame memory for use by the 3B20D Central Control and Input/Output system. The memory system is designed using a collection of high-speed, static and dynamic memory devices and appropriate logic controllers. In addition to providing basic on-line storage for program text and data, the memory system provides hardware assistance for virtual-to-physical address translation, access protection, memory resource arbitration, and performance enhancement utilizing a high-speed cache memory. The technology used in implementing these functions includes state-of-the-art 64K dynamic random access memory devices and high-speed TTL-compatible gate-array integrated circuits.

I. INTRODUCTION

The memory system associated with the 3B20D Processor[1] includes a 16-megabyte memory, a high-speed cache memory, and hardware assistance for the virtual-to-physical address translation process, access protection, and memory resource arbitration functions.[2] The memory system utilizes high-speed, static and dynamic memory devices and appropriate logic controllers.

The block diagram shown in Fig. 1 highlights the major components and interconnections of the 3B20D Memory System. The diagram indicates the memory system related control, address, and data paths, including the interconnection to the fully duplicated system. Internal central control data paths associated with the Store Address Translator (SAT) are not shown.

As indicated, the 3B20D Memory System is comprised of a 16-

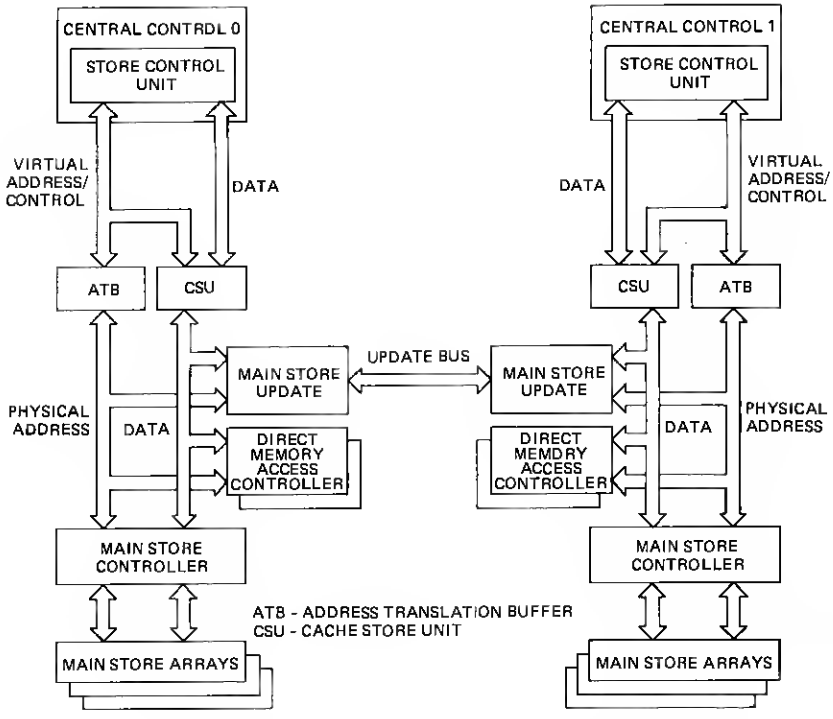

Fig. 1—Block diagram of the 3B20D Processor memory system.

megabyte main store, a Main Store Update (MASU) circuit providing interconnection control between the duplicated memory systems, a store address translator, and an optional Cache Store Unit (CSU). The subsystems that use the memory system are the central control and the Direct Memory Access Controller (DMAC). Memory operations are initiated by either of the duplicated central controls or DMACs. In the DMERT environment, one CC/DMAC combination has control of the system and initiates all memory operations. The MASU circuit performs write operations to both duplicated main stores. In this way, both main stores are kept up to date with currently executing programs, data, and I/O transactions. If a central control switch is needed, it can be accomplished quickly since the off-line central control can immediately use a fully updated main store.[3]

The 3B20D Processor supports memory management that allows programs to be written using virtual addresses without regard to where they actually reside in memory. An address translation hardware assistance circuit, the SAT, is provided in the 3B20D. The SAT provides high-speed access to the most recently used address-translation and access parameters. It serves as a cache memory for the

translation parameters that apply to the software processes executing in the central control. The same address-translation mechanism implemented by the SAT is used by the DMAC.[4,5] Thus, processes executing in the central control can share virtual address spaces with peripheral devices communicating through the DMAC. In addition, once this address-translation mechanism is established by the operating system, the central control or the peripheral devices may initiate memory operations independently.

The remainder of this paper discusses, in more detail, the major components of the memory system. The discussions cover the operational aspects of the design, self-checking, error reporting, error recovery, and diagnostic features[6] provided in the memory system design. Other topics addressed include performance, reliability, device technology, and environmental considerations.

II. STORE ADDRESS TRANSLATOR

In any computer system, the main memory is an expensive resource and has to be efficiently managed for improved system performance. Because the 3B20D Processor supports a multiprogramming environment, the memory may be shared by several user programs, each having access to the full virtual memory spectrum. Hence, a mechanism for relocation and protection of a user's address space (text and data) is required. The 3B20D translates the virtual addresses used within the processor and I/O subsystem to a real or physical address used within the main store. To reduce the overhead associated with the address-translation and protection mechanism, the 3B20D uses an SAT that contains a high-speed address cache called the Address Translation Buffer (ATB).

2.1 Address space partitioning

For the purpose of sharing, relocating, and protecting, the 24-bit address space is partitioned into "segments." A segment is a contiguous block of sequential virtual addresses the size of which may range from 1 byte to 128K bytes. To reduce memory breakage due to fragmentation, each segment is further partitioned into 2K-byte blocks called "pages." A segment has up to 64 pages and an address space can have up to 128 segments. The virtual address generated by the processor is divided into a 7-bit segment field, a 6-bit page field, and an 11-bit byte offset (displacement) as shown in Fig. 2.

2.2 Address translation process

The SAT translates the 24-bit virtual address to a 24-bit physical address used by the CSU and the main store. This translation is conceptually a two-step table-lookup operation and is controlled by

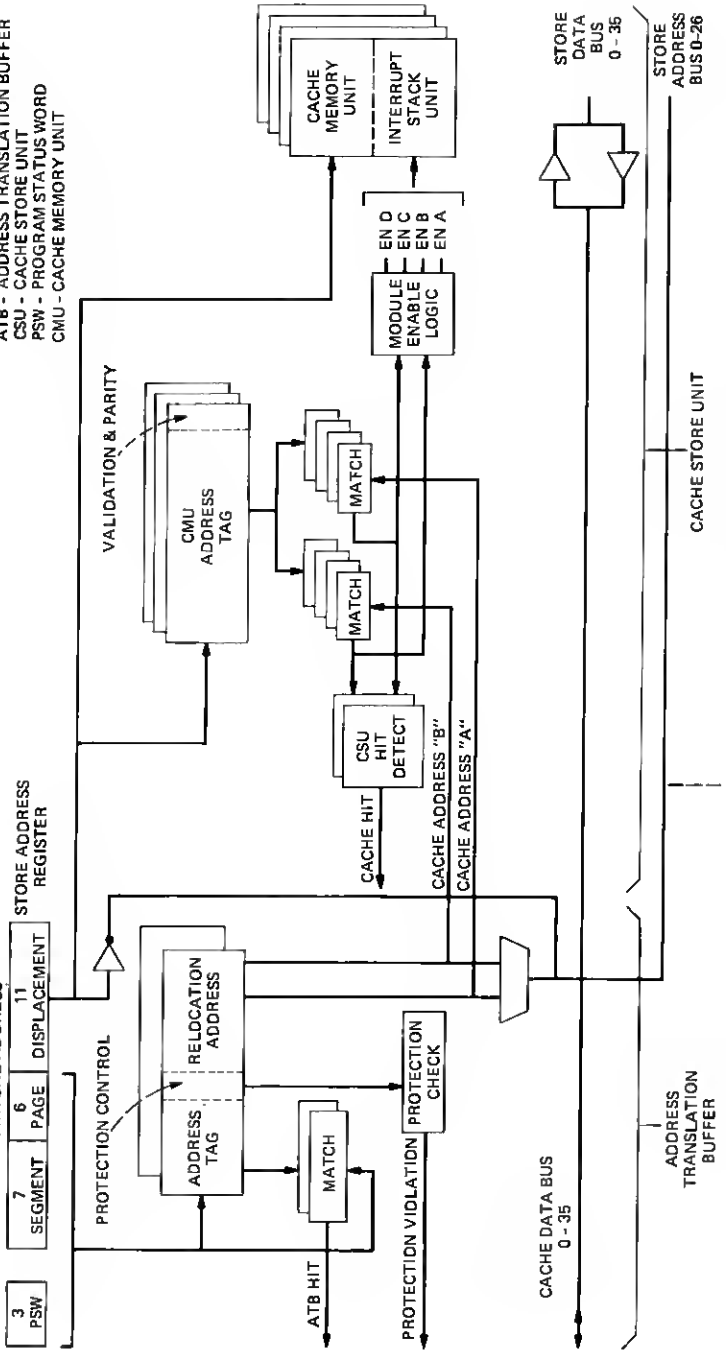

Fig. 2—Block diagram of the 3B20D Processor store address translator and cache store unit.

the Program Status Word (PSW), the segment base register (SBR), and two main store resident tables (the segment table and the page table). The resident operating system manages these registers and tables. Recently used translation information is kept in the high-speed ATB and is accessed concurrently with the CSU.

2.3 Segment base register

The currently executing process's[2] address space is defined by the SBR. The SBR is a 32-bit register that contains the physical address pointer to the beginning of the process's segment table and its length (which is equal to the number of segments allocated to the process). There are eight such SBRs in the processor to accommodate eight independent address spaces at any time. These eight address spaces are assigned by software, and in a DMERT environment, a minimum subset of four is allocated to the kernel, a kernel process, a supervisor process, and a user process. The fields in an SBR are allocated as follows:

(*i*) Segment table address (22 bits)—This field points to the beginning of the segment table in main store.

(*ii*) Unused (3 bits)—This field is presently not used.

(*iii*) Segment limit (7 bits)—This field designates the length of the segment table.

2.4 Segment table

The segment table (one exists for each process) contains a descriptor for each segment of the process. Each entry is 4 bytes long and resides on a full word boundary in the main store. It is partitioned into three fields as follows:

(*i*) Page table address (22 bits)—This field points to the beginning of the page table.

(*ii*) Page table length (6 bits)—The length of the page table is one more than the value in this field.

(*iii*) Protection bits (4 bits)—Three of these bits indicate whether the segment is readable, writable, or executable. One bit indicates the validity of the entry.

2.5 Page table

The page table (one exists for each segment) contains a descriptor for each page in the segment. Each entry is 4 bytes long and resides on a full word boundary in the main store. The entry has four fields that indicate:

(*i*) Relocation Address

(*ii*) DMA usage

(*iii*) Protection
(*iv*) Unused.

The process of virtual to physical address translation is described in the following sections.

2.5.1 Segment table lookup

The segment field of the virtual address is first compared against the segment limit field of the SBR to establish whether the addressed entry is within the table. If the segment limit is less than the segment field, a "segment length error" exception is recognized and control of the processor is transferred to handle the exception. If there is no such error, the segment field of the virtual address and the segment table address in the SBR are used to index into the segment table in the main store and access an entry.

The fetched segment table entry is checked for validity. If the entry is invalid, control is transferred to exception handling microcode. If the entry is valid, the page field of the virtual address is compared against the page table length field in the segment table entry. If the value of the page field is greater than the maximum number of pages in the segment, the control is transferred to a microcode routine to handle the error. In case of no such error, a page table lookup is initiated.

2.5.2 Page table lookup

The contents of the page-field of the virtual address and the page table address field of the fetched segment table entry are used to index into the page table to fetch a page table entry. The protection bits obtained from the segment table entry and the page table entry are ANDed, and a check is made for a possible protection violation. In case of such an error, an error-handling routine is initiated. If there is no protection violation, then the physical address is generated by concatenating the relocation address field of the fetched page table entry and the byte offset field of the virtual address.

2.6 Address translation buffer

The Address Translation Buffer (ATB) is provided to reduce the overhead associated with the address-translation mechanism. The ATB is capable of holding 128-page table entries for eight different address spaces (for a total of 1024 entries) and is organized as a two-way set associative memory. When the processor initiates a store access, the ATB checks whether the corresponding physical address is available in its memory. If the physical address is available (called a "hit"), it is sent to the main store. But, if the physical address is not available (called a "miss"), an "ATB Miss" processing microroutine is

initiated which does the segment and page table lookups and loads the obtained physical address into the ATB. The access is then restarted. Subsequent access to the same page results in hits.

2.7 ATB control bits in PSW

The PSW contains a field that provides control for the memory management. The ATB functions can be enabled or disabled. When disabled, address-translation and protection check functions are disabled, and the virtual addresses are directed to the CSU and the main store as physical addresses.

The PSW also controls process communication capabilities between virtual address spaces. The PSW designates the Primary Segment Base Register (PSBR) and the Secondary Segment Base Register (SSBR). Each of these registers defines a virtual address space. Generally, the PSBR is used to select one of the eight blocks of the ATB for address translation. Under PSW control, however, the SSBR can be used for read, write, or both read and write memory operations. All instruction fetches use the PSBR irrespective of the contents of the PSW. Special instructions are provided to manipulate the PSW. This hardware feature can be used to move data between two address spaces very efficiently.

2.8 ATB operation

Figure 2 shows the block diagram of the ATB. When an access is initiated by the processor, the ATB is accessed using 3 bits of the PSW (selecting one of eight address spaces), low order 3 bits of the segment field, and low order 3 bits of the page field of the virtual address. The address tag fields of the ATB are matched against the corresponding bits of the virtual address. Simultaneously, the two relocation address fields are directed to the cache. A hit is generated in case of a successful match.

If a hit is detected, then a check is done to see whether the access is allowed on that page. The processor and the cache are informed in the case of any protection violation.

If a miss is detected, the cache ignores the relocation addresses. An "ATB Miss" microroutine is initiated by the processor and the new ATB entry is loaded into one set of the ATB using a defined replacement algorithm. The access is then restarted.

As mentioned before, the ATB is capable of handling translations for eight tasks at any instant. But, when a new task is allocated to a block of the ATB, the entries associated with the previous task have to be invalidated. A dedicated 8-bit counter is provided to do this invalidation with minimum microcode overhead.

The address translation mechanism provided by the SAT is com-

patible with that provded by the DMAC. In this way, processes executing on peripheral controllers can share virtual address spaces with processes executing on the central control.

2.9 Redundancy

The ATB hardware is completely self-checked. Parity is checked over the Store Address Register (SAR), the ATB entries and the ATB related bits in the PSW. Parity bits are regenerated for the physical addresses to be checked at the cache and main store. The hit and protection check logics are duplicated and matched, and can be exercised under maintenance control. In case of any hardware faults in the circuit-pack, the access is aborted and control is transferred to a fault-handling routine. The ATB memory can be written and read over the source and destination buses of the processor.

III. CACHE STORE UNIT

The Cache Store Unit (CSU) reduces the effective access and cycle time of store operations for the 3B20D Processor. Combining a relatively small, high-speed "cache" memory with a large main store results in a system with an average access time approaching that of the high-speed cache but with the low cost per bit and storage capacity of the Main Store.

The concept of a cache takes advantage of a general programming characteristic of locality of reference. Most references to memory tend to be highly localized or clustered into small groups at any given time, and regions tend to change relatively slowly during the course of program execution. Thus, a relatively small, high-speed CSU contains the most often used words from the main store and thereby reduces the average access time of the reference.

3.1 Organization

The CSU is organized into two sections: an interrupt stack section and a cache section. Since the main store contains a much larger storage capcity than the CSU, a mapping function is required to compress the main store address range into the much smaller CSU. The compression is achieved by adding a tag which contains the physical page address to each word of cache data storage. To increase the probability that a main store word is in the CSU (a "hit"), the cache is organized as a four-way set-associative memory. Each one of the four cache modules contains 2K bytes of high-speed storage. The interrupt stack section also contains 8K bytes of memory.

Since the cache is four-way set-associative with the main store, there is a one for one correspondence to the page offset address but full associativity for the page portion of the address. Thus, word 0 from

any of the four cache modules may correspond to word 0 from any page in main store.

3.2 Operation

Since the cache section of the CSU functions as an associative memory, a match search in the cache is performed in parallel with the translation of the virtual-to-physical address by the ATB. This hardware parallelism is illustrated in Fig. 2. An access causes the low 11 bits of the virtual address to select a unique page offset on each of the four cache modules. The page portion of the virtual address (high 13 address bits) is translated by the two-way set-associative ATB. Each of the four cache tag modules is matched to the two translated physical page addresses. The ATB will indicate to the cache which of the two translated addresses is valid. If one of the four cache tag modules matches this translated page address, the CSU will generate a hit signal to the CPU and gate the associated word onto the cache data bus.

Functionally, the CSU interconnects the CPU and main store. The CSU connects to the CPU via the cache address bus, cache data bus, and control leads. Because the CSU interconnects the CPU and main store, DMA transfers to main store will not prevent the CSU from being accessed by the CPU. Thus, CPU contention for the main store is reduced since most references will result in a CSU hit.

During system initialization, all locations in the CSU are invalidated. When the CPU references the memory system and a cache miss results, the referenced word is automatically copied into the CSU. If a word needs to be copied from the main store and all four cache modules contain valid data, a random replacement algorithm selects the cache group in which a word will be replaced. Once the cache has been initialized by the system, the operation is transparent to the software with the exception of enhanced performance.

The CSU contains arbitration and sequencer control logic to automatically update its data and arbitrate between CPU and DMA write operations. When the DMA writes into the main store the CSU checks if the DMA reference is in the cache. If the word is in the cache, it is automatically invalidated by the CSU sequencer. While the CSU is checking for a DMA write hit, the CSU indicates a busy condition to the CPU. DMA reads from the main store do not result in any operation from the CSU.

3.3 Interrupt stack

The CSU also contains an 8K-byte high-speed memory which is used by the CPU to reduce the interrupt response time when the CPU is in the kernel operating mode. When the CSU is in the interrupt

stack mode, the CSU will generate 100-percent hits for both read and write operations.

3.4 Software support

The CSU is designed to support the software organization of the DMERT operating system and the "C" programming language. Call and Return are frequent operations with the C language. The CSU has built-in hardware algorithms to function as a high-speed data stack. A stack write operation (CSAV) will force the data to be copied into both the cache and main store. A stack read operation (CRET) will cause the data to be read from the cache and then invalidated. This location in the cache is then available for use by new data. The data stack is part of the cache section and is totally separate from the interrupt stack.

3.5 Self-checking

The self-checking philosophy of the CSU is to provide immediate detection of faults that cause errors and nonimmediate detection of faults that affect the performance of the CSU. The CSU has built-in self-checking hardware that monitors the operation of the CSU. Faults detected by the self-checking hardware include CSU sequencer errors, multiple writes into the CSU, multiple cache hits, accessing errors in the tag memories, and cache write errors.

A multiple hit is an example of a catastrophic fault since it would result in the contents of two or more cache words being ORed onto the cache data bus. This type of failure is prevented by duplication of the hit logic and by providing multiple hit detectors that monitor the hit logic and the module enable logic. The cache also generates and checks parity over the tag and address bits.

The CSU also provides diagnostic access to the "internals" of the circuit. When the CSU is configured in the maintenance mode, the normally associative memory tages are configured to function as conventional RAM memories. In addition, special access is provided to the counters, CSU sequencer, and various status bits.

IV. MAIN STORE

4.1 Configuration

Physically the main store can consist of one or two modules. Each module contains one Main Store Controller (MASC) and up to sixteen Main Store Arrays (MASA). The central control can directly address 16 megabytes of text or data.

The preproduction design of the main store was based on the initial use of 16K Dynamic Random Access Memory (DRAM) devices. The design was organized to allow evolution to higher-density devices. The

production design initially used a Western Electric 64K DRAM[7] that utilized two power supply voltages. The current design uses a newer Western Electric 64K DRAM that permits denser packaging. This design is implemented using a single circuit pack main store controller and a 1-megabyte MASA circuit pack. Thus, all 16 megabytes of addressable memory are contained in one module.

4.2 MASU operational aspects

The MASU circuit provides control of main store bus communication between duplex Control Units (CU), allowing the MAS in the standby CU to be kept up to date. The MASU also controls the use of the main store buses by the central control, DMA, and other CUs. The MASU in the active CU in a duplex configuration gives the highest priority to the central control followed by DMA 0, DMA 1, and operations from the other CU, respectively.

Communication with the Main Store is over the address bus, data bus and the command bus. The command leads indicate whether the operation to be performed is a write, read, clear, byte, halfword, or maintenance operation. Valid operations to the MAS include:

(*i*) Write full word
(*ii*) Write halfword/byte
(*iii*) Read word
(*iv*) Read and clear fullword
(*v*) Read and clear halfword/byte
(*vi*) Maintenance write (nonmemory operation)
(*vii*) Maintenance read (nonmemory operation).

The MASU communicates asynchronously with the MASC by the use of the Store Go signal (SGO) and Store Complete signal (SCM). Prior to issuing the SGO to the MASC, the MASU issues an address and data bus enable to the requesting unit with the highest priority. The MASU then issues the SGO to the MASC. The MASC upon receiving the SGO begins a timing sequence that selects the addressed MASA and issues a GO signal (GOI) to the MASA. The selected MASA then allows data to be read or written at the specified address depending upon which main store operation was decoded by the MASC.

The MASC performs various error checks during the timing sequence to ensure tbe integrity of the MASC and MASA. In the event an error does occur, tbe MASC sends an error signal to the requesting unit. Depending upon the operation being performed and the state of the error sources, the MASC at a specific point in the timing sequence issues an SCM to the MASU to indicate that the operation has been completed. The MASU can then grant the main store buses to the next requesting unit with the highest priority.

4.3 Self-checking

The 3B20D main store was designed with a significant amount of self-check circuitry. The partitioning of the circuitry on the circuit packs also was designed to improve fault detection. For example, the data bus transceivers on the memory array have only one bit from each parity field partitioned in each device. In this way a fault affecting either one of the bits or all the bits in the device will be detected by a simple parity check. Similarly the address, RAS, and CAS drivers for the memory devices on the MASA were partioned so that a fault associated with one of these drivers would affect at least two bits in each of the data-parity fields. By affecting more than one byte the probability of detecting the error by failing a byte-parity check increases.

To ensure bus integrity the MASC checks the address, data, and command bus parity between the MASC and the MASU. If the address parity check fails during a write cycle to the memory, circuitry in the MASC prevents the writing of data into the addressed memory location. Thus, invalid information will not be written into the memory. If the other parity checks fail, an error is signaled but an undesired memory operation may take place. The MASC internally monitors the timing sequencer and refresh address counters to ensure that they are functioning properly. Circuitry also checks the SGQ and SCM signals between the MASC and MASU to ensure that "handshaking" that takes place between the two units is functioning properly. The MASC by checking four selected responses from the MASA also can check communication to the MASA. The MASC can determine from the select responses whether an MASA was selected or not, and it can also detect if more than one MASA responded.

4.4 Error correction

The 3B20D main store performs error detection and correction. Error correction circuitry on the MASC and Error Correction Coding (ECC) of data in the MASA result in the correction of all single bit errors. This circuitry also flags all double and detectable multi-bit errors.

In the 3B20D central control, byte parity is maintained over each byte of the data word. The main store uses the existing byte-parity bits in a modified form of the Hamming code.[8] By adding four additional ECC bits (in addition to the byte-parity bits) the main store can then perform single-bit correction and double-bit failure detection. When presenting data to the system the MASC maintains byte parity by gating the byte parity bits to the MASA from the MASU, and vice versa. On the MASC five gate arrays are used to implement the error-checking and correcting circuitry. One gate array code is used for each

of the four bytes. The fifth gate array code is used for the additional four ECC bits.

The MASC issues an error signal when it detects a location in MAS in need of correction. The MASC presents the corrected data to the system but will not automatically rewrite correct data at the location. The actual rewriting of data is handled by the software error interrupt handler. In the event of a noncorrectable error, the error interrupt handler reads the data from the standby CU—if it is in a duplex configuration—and uses that data to rewrite the faulty location.

4.5 Diagnostics

The 3B20D MASC provides maintenance access to a significant portion of the circuitry in the main store. This maintenance access is used by microcode to initialize the MASC and by diagnostics to test the functional operation of the MAS.

The MASC issues various maintenance commands within the MAS when the maintenance command, address, and SGO signal are presented to the MAS. The maintenance operation is decoded off tbe address and is only valid for one MAS cycle. By using an address decoded maintenance command, the state of certain data bits can be latched, providing the latched maintenance state until cleared. The MASC under diagnostic control uses the various maintenance commands and states to perform the following operations:

(*i*) Control address loop-around (address returns on data bus)
(*ii*) Control the refresh circuitry
(*iii*) Control the error correction circuitry
(*iv*) Control the error detection/reporting circuitry.

By using the maintenance capability, the diagnostics can verify the integrity of the bus structure, the MASA, and the MASC. The data integrity of each MASA is tested by performing a series of data pattern tests on each MASA equipped.

V. ACKNOWLEDGMENTS

The authors would like to thank D. J. Matter, C. M. Narayanan, and D. M. Trampel for their assistance in the preparation of this manuscript and their contributions to the 3B20D memory system design. D. R. Draper and W. A. Read also have made significant contributions to the main store design.

REFERENCES

1. M. W. Rolund, J. T. Beckett, and D. A. Harsm, "3B20D Central Processing Unit," B.S.T.J., this issue.
2. M. E. Grzelakowski, J. H. Campbell, and M. R. Dubman, "DMERT Operating System," B.S.T.J., this issue.

3. R. C. Hansen, R. W. Peterson, and N. O. Whittington, "Fault Detection and Recovery," B.S.T.J., this issue.
4. A. H. Budlong and F. W. Wendland, "3B20D Input/Output System," B.S.T.J., this issue.
5. R. E. Haglund and L. D. Peterson, "3B20D File Memory Systems," B.S.T.J., this issue.
6. J. L. Quinn, R. L. Engram, and F. M. Goetz, "Diagnostic Tests and Control Software," B.S.T.J., this issue.
7. R. P. Cenker, D. G. Clemons, W. R. Huber, J. B. Petrizzi, F. J. Procyk, G. M. Trout, "Fault Tolerant 64K Dynamic Random Access Memory," IEEE Transactions of Electronic Devices, *ED26*, No. 6 (June 1979), pp. 853–60.
8. R. W. Hamming, *Coding and Information Theory*, Englewood Cliffs, N.J.: Prentice-Hall, 1980.